CLAIMS:

1. A method for memory allocation for a multithreaded processor, comprising:

obtaining threads;

creating sets from the threads according to thread type;

allocating memory space for each thread in a set responsive to the thread type of the set; and

allocating memory spaces for data buffers accessible by the threads.

2. The method according to claim 1, wherein a location in a data buffer is written to by a thread and read from by another thread.

3. The method according to claim 1, further comprising:

determining if a read-after-write hazard exists for a location in a memory space for a thread.

4. The method according to claim 1, wherein a position in a data buffer is determined by a thread.

5. The method according to claim 1, wherein a data buffer is associated with a buffer identifier.

6. The method according to claim 5, wherein a memory location address in the data buffer is determined using the buffer identifier and position coordinates associated with a sample.

7. The method according to claim 1, wherein a size of a data buffer is different than a size of another data buffer.

8. A method for accessing a memory resource for a plurality of threads simultaneously executable in a graphics processor responsive to a graphics program module, comprising:

processing a first sample by a first thread; and

processing a second sample by a second thread;

wherein a first memory space is accessed only by the first thread during processing of the first sample by the first thread, a second memory space is accessed only by the second thread during processing of the second sample by the second thread, and a third memory space is accessed by the first thread and the second thread.

9. The method according to claim 8, wherein the third memory space stores data buffers.

10. The method according to claim 8, wherein the first thread determines a position in a data buffer.

11. The method according to claim 8, further comprising:

determining if a read after write hazard exists for a location in a memory space accessed by a thread.

12. The method according to claim 8, further comprising:

writing sample data to a location in a data buffer by the first thread; and

reading the sample data from the location in the data buffer by the second thread.

13. The method according to claim 8, wherein the data buffer in the third memory space is associated with a buffer identifier.

14. A computer program product having a computer readable medium having computer program instructions recorded thereon, said computer program product comprising:

instructions for determining a first set of threads of a plurality of threads, each thread in the plurality of threads being associated with a graphics program module executing on a graphics processor;

instructions for allocating a first memory space in a memory resource to each thread in the first set of threads, the first memory space being reserved for the thread to which the first memory space is allocated; and

instructions for allocating a second memory space in the memory resource to a data buffer, the data buffer being accessible by each thread in the plurality of threads.

15. The computer program product of claim 14, wherein a thread writes a memory location in the second memory space and another thread reads another memory location in the second memory space.

16. The computer program product of claim 15, wherein the other thread reads the other memory location in the second memory space after determining a read-after-write hazard does not exist for the other memory location in the second memory space.

17. The computer program product of claim 14, further comprising:

instructions for allocating additional memory spaces in the memory resource to additional data buffers, each data buffer being accessible by each thread in the plurality of threads.

18. The computer program product of claim 17, wherein the data buffer is associated with a unique buffer identifier and the additional data buffers are each associated with unique buffer identifiers.

19. A computing system comprising:

a memory resource;

a graphics processor coupled to the memory resource for executing one or more graphics program modules;

a central processing unit (CPU) coupled to the memory resource and the graphics processor, wherein the CPU or the graphics processor:

determines a first set of threads from a plurality of threads simultaneously executable in the graphics processor;

allocates a memory space to each thread in the first set of threads to respectively reserve the memory space; and

allocates an additional memory space accessible by each thread in the first set of threads to one or more data buffers to respectively reserve the additional memory space.

20. The computing system of claim 19, further comprising:

an address unit coupled to the graphics processor and the CPU, wherein the address unit:

receives an access command from a thread in the first set of threads to access the memory space; and

determines if a read-after-write hazard exists prior to accessing the memory space.

21. The computing system of claim 20, wherein the address unit uses an operation command and address request information to determine if a read-after-write hazard exists.

22. The computing system of claim 20, wherein the address unit uses a thread identification number and a memory location offset to determine if a read-after-write hazard exists.

23. The computing system of claim 19, further comprising:

an address unit coupled to the graphics processor and the CPU, wherein the address unit:

receives an access command from a thread in the first set of threads to access the additional memory space; and

determines if a read-after-write hazard exists prior to accessing the additional memory space.

24. The computing system of claim 19, wherein each data buffer in the additional memory space is associated with a unique buffer identifier.

25. The computing system of claim 24, wherein the address unit uses the buffer identifier and sample position coordinates to determine if a read-after-write hazard exists.